

vi 사용법 익히기

* 학습목표

- 유닉스에서 사용하는 편집기의 종류를 알아본다.
- 대표적인 화면 편집기인 vi의 사용 방법을 익힌다.
- vi의 환경설정 방법을 익힌다.

01. 유닉스 편집기

02. vi의 사용 방법

03. vi의 환경설정

요약

연습문제



유닉스 편집기

① 유닉스 편집기의 종류

유닉스도 윈도우처럼 편집기를 사용해 파일을 작성하거나 수정한다. 즉, 편집기를 사용해 반복적인 편집 작업을 자동으로 처리하고 간단한 문서나 프로그램을 작성한다. 유닉스 편집기는 다른 운영체제처럼 기본으로 제공되거나 따로 설치해야 하는데, 크게 행 단위 편집기와 화면 단위 편집기로 구분할 수 있다. 행 단위 편집기는 한 번에 한 행씩만 작성하거나 수정할 수 있고, 화면 단위 편집기는 윈도우에서 일반적으로 보는 것처럼 화면 단위로 전체 내용을 보면서 커서를 이동하며 작업할 수 있다.

[표 4-1] 유닉스 편집기 종류

구분	종류
행 단위 편집기	ed, ex, sed
화면 단위 편집기	vi, 이맥스

ed는 유닉스 초기의 행 단위 편집기로, 사용이 불편해 지금은 거의 사용하지 않는다. ex도 행 단위 편집기이지만 단독으로 사용하기보다는 vi와 연결해 vi를 더욱 강력하게 하는 다양한 기능을 제공한다. sed는 스트림 편집기로 일반 편집기와 달리 지시된 명령에 따라 파일의 내용을 일괄적으로 바꿔 출력해 준다. sed는 셸 스크립트를 작성할 때 파일을 읽어들이어 편집하는 기능을 수행하기 위해 많이 사용된다.

이맥스(Emacs)는 화면 단위 편집기로, 그 종류가 다양한데 그 중 GNU 이맥스가 가장 유명하다. GNU 이맥스는 무료로 배포되고 있으며, 보통 기본적으로 설치되지 않아 별도로 설치해야 한다. 이맥스의 가장 큰 장점은 이맥스를 사용해 파일을 편집하면서도 유닉스의 여러 기능들을 그냥 사용할 수 있다는 것이다. 예를 들어 이맥스로 편집하는 도중에 이메일을 보내거나 받을 수도 있고, 셸을 실행하거나, 파일을 복사하고, 삭제하는 등의 작업을 실

행할 수 있다. vi는 유닉스에서 기본적으로 제공되는 화면 편집기이다. vi는 ex 편집기를 바탕으로 만들어졌기 때문에 ex 편집기의 명령을 그대로 사용할 수 있다. 취향에 따라 vi를 사용하는 사람도 있고 이맥스를 사용하는 사람도 있다. 이맥스를 주로 사용하는 이유는 이맥스가 가진 강력하고 풍부한 기능을 선호하기 때문이고, vi를 선호하는 경우는 vi에서 사용하는 명령들이 매우 단순해 빠른 편집이 가능하기 때문이다.

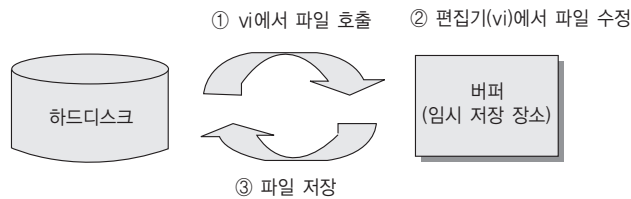
이 장에서는 유닉스에 기본적으로 설치되어 있는 편집기인 vi의 사용법을 살펴본다. 유닉스에서 동작하는 편집기는 공통점이 있다. 따로 익히지 않으면 절대 사용할 수 없다는 것이다. 책 없이도 간단한 편집을 할 수 있는 워드나 아래 한글과는 근본적으로 다르다. vi의 사용 방법을 아느냐 모르느냐는 자량의 수준의 아니라 시간과 관련있다. 그러므로 이 장에서 학습하는 내용을 반드시 숙지하고, 욕심을 부린다면 매일 시간을 투자해 완전히 정리된 vi를 조금씩 학습하라고 권하고 싶다.

② 모드형과 비모드형 편집기

대표적인 유닉스 편집기인 vi는 모드형이고 이맥스는 비모드형 편집기이다. 비모드형에서 a는 항상 “a”라는 글자를 뜻한다. 그럼, a는 당연히 “a”지 “b”도 될 수 있다는 말인가? 바로 그 점이 모드형 편집기와 비모드형 편집기의 차이이다. 모드형 편집기인 vi에서 j는 “j”라는 글자 자체를 말하기도 하고 화면에서 아래로 가는 방향키가 되기도 한다. 그러므로 이맥스에서는 **Ctrl** 키와 문자를 결합해 명령문을 만들지만 vi에서는 명령 모드에서 입력한 모든 것이 명령으로 처리된다.

2 vi의 사용 방법

vi는 기존 파일을 편집할 경우 [그림 4-1]과 같이 하드디스크에서 파일을 편집기로 불러온 뒤 파일을 편집한다. 이 때 vi로 불러들여 편집한 파일은 메모리 버퍼에서만 바뀐 것이지만 아직 하드디스크에 저장된 것은 아니다. 사용자가 파일 저장 명령을 주어야 비로소 편집된 내용이 하드디스크에 저장된다.



[그림 4-1] vi의 동작 구조

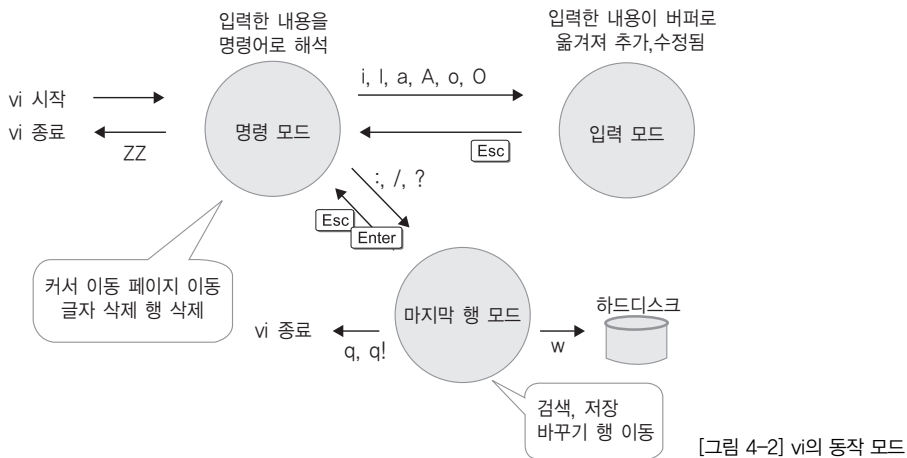
1 vi의 동작 모드

vi의 사용법을 본격적으로 익히기 전에 모드형 편집기인 vi의 동작 모드에 대해 알아보자. vi의 모드는 입력 모드와 명령 모드, 마지막 행 모드로 구분된다(마지막 행 모드를 명령 모드에 포함시켜 크게 입력 모드와 명령 모드로만 구분하기도 한다). 입력 모드는 실제로 내용을 입력할 수 있는 모드이고, 명령 모드와 마지막 행 모드는 글자나 행의 삭제, 검색, 저장 등의 기능을 수행한다.

vi에서는 입력 모드와 명령 모드를 오가며 작업을 한다. 윈도우에서 많이 사용하는 문서 편집기인 아래 한글과 비교해 보자. 한글도 내용을 입력하는 입력 기능과 입력된 내용을 편집하는 명령 기능이 있다. 한글에서 명령 기능이란 **[Alt]** 키나 **[Ctrl]** 키와 같은 특수키와 함께 다른 키를 입력하면 이것을 입력으로 처리하지 않고 명령으로 처리하는 것을 의미한다. 즉, a는 입력으로 처리하고, **[Ctrl]** + a는 명령으로 처리하는 것이다. 그런데 vi에서는 동일키로 입

력도 하고 명령도 처리한다. 같은 a를 입력해도 입력 모드일 때는 a라는 글자가 입력되지만 명령 모드에서는 문자 추가 명령이 실행된다. 이것이 처음 vi를 접하는 사람들이 어렵게 생각하는 부분이기도 하다. 하지만 조금만 익숙해지면 아주 편리하다는 것을 알 수 있다.

[그림 4-2]를 보면서 정리해 보자. vi를 시작하면 자동으로 명령 모드가 되어 이때 입력하는 키는 모두 명령으로 해석된다. 명령은 대소문자를 구별한다. 명령 모드에서 입력 모드로 가기 위해서는 i, I, a, A, o, O 중 하나를 입력하면 된다. 입력 모드가 되면 지금부터 입력하는 키는 내용으로 이해해 버퍼에 저장된다. 입력 모드에서 다시 명령 모드로 가기 위해서는 [Esc] 키를 입력한다. 검색이나 바꾸기 같은 특별한 명령을 수행하기 위해서는 마지막 행 모드로 가야 하는데 이는 명령 모드에서 :, /, ? 중 하나를 입력하면 된다. 마지막 행 모드에서 명령을 입력하고 실행하려면 반드시 [Enter] 키를 입력해야 한다. 그리고 마지막 행 모드에서 명령 입력 도중 [Esc]를 입력하면 명령 모드로 돌아간다.



2 vi의 시작과 종료

vi 명령의 기본 기능과 사용법은 다음과 같다.

vi 명령

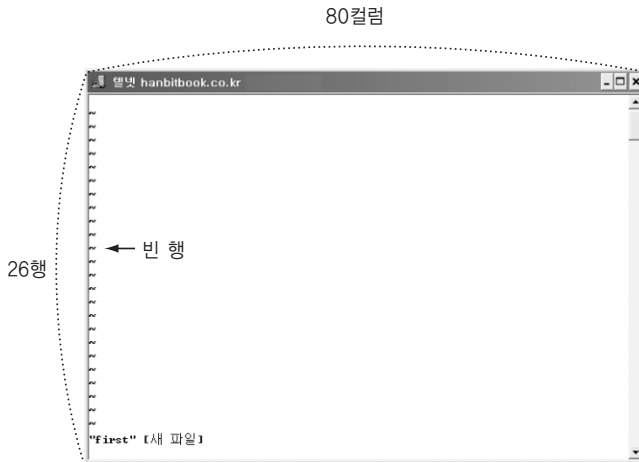
- 기능 ◉ 지정한 파일을 편집한다. 파일명을 지정하지 않으면 빈 파일이 열리고 이 빈 파일의 파일명은 별도로 지정할 수 있다.
- 형식 ◉ vi [파일명...]

vi 시작하기

vi를 시작할 때 파일명을 지정할 경우 기존에 있던 파일이면 해당 파일이 열리고, 없는 파일이면 빈 파일이 열린다. 그리고 파일명을 지정하지 않으면 빈 파일이 열린다. 그리고 이 경우 파일명은 나중에 마지막 행 모드에서 지정할 수 있다.

```
$ vi first → first라는 파일이 열린다.
$ vi      → 빈 파일이 열린다.
```

vi로 파일을 열면 나타나는 단말기의 전체 화면을 편집 화면으로 사용한다. 단말기가 기본으로 설정된 경우 화면은 보통 26행 80컬럼이다. 그리고 각 행의 처음에 나타나는 틸드(~)는 빈 행임을 나타내는 표시이다.



[그림 4-3] vi의 초기 화면

vi 종료하고 파일 저장하기

편집을 완료하고 vi를 종료하기 위해서는 마지막 행 모드에서 명령을 입력해야 한다. 보통의 편집기처럼 작업한 내용을 저장하지 않고 종료할 수도 있고, 다른 파일명을 지정해 저장할 수도 있다. 명령 모드에서 파일 저장과 종료를 동시에 수행할 수 있는 명령키가 하나 있는데 바로 **[Shift]+zz** 즉 대문자 ZZ이다. 이 명령은 파일을 저장한 후 vi를 바로 종료시킨다. vi를 종료시키기 위한 다른 명령키들은 마지막 행 모드에서 실행할 수 있다. 마지막 행

모드로 가기 위해서는 : 키를 먼저 입력한다. vi 종료와 파일 저장을 위해 사용할 수 있는 명령키들을 정리하면 [표 4-2]와 같다.

[표 4-2] vi 종료와 저장 명령키

명령키	기능
:q	vi에서 작업한 것이 없을 때 그냥 종료한다.
:q!	작업한 내용을 저장하지 않고 종료한다.
:w [파일명]	작업한 내용을 저장만 한다. 파일명을 지정하면 새 파일로 저장한다.
:wq, :wq!	작업한 내용을 저장하고 vi를 종료한다
ZZ([Shift]+zz)	작업한 내용을 저장하고 vi를 종료한다.

③ 입력 모드로 전환

vi를 시작하면 기본적으로 명령 모드이다. 입력을 위해서는 [그림 4-2]에서 살펴본 바와 같이 명령 모드에서 입력 모드로 전환해야 한다. 입력 모드로 전환하기 위해서는 다음 중 하나의 명령키를 이용한다.

[표 4-3] vi 입력 모드 전환 명령키

명령키	기능
i	커서 앞에 입력한다(현재 커서 자리에 입력).
a	커서 뒤에 입력한다(현재 커서 다음 자리에 입력).
o	커서가 위치한 행의 다음 행에 입력한다.
I	커서가 위치한 행의 첫 컬럼으로 이동해 입력한다.
A	커서가 위치한 행의 마지막 컬럼으로 이동해 입력한다.
O	커서가 위치한 행의 이전 행에 입력한다.

i 명령키를 사용한 입력 모드 전환

먼저 i 키의 사용 예를 살펴보자. vi를 실행한 뒤 명령 모드에서 i 명령키를 입력하면 아래와 같이 내용을 입력할 수 있다. 커서는 x자 위에 있게 된다. 이 때 i 명령키를 두 번 입력하면 i가 화면에 나오므로 주의해야 한다.

```
unix vi test      → [Enter] 키를 입력하면 다음 행으로 이동한다.
I like unix      → [Esc] 키를 입력하면 명령 모드로 바뀐다.
  x
~
~
```

o 명령키를 사용한 입력 모드 전환

명령 모드 상태에서 현재 커서가 위치한 행의 다음 행에 글자를 입력하려면 o를 입력한다. 그러면 다음과 같이 현재 커서 위치의 아래 행으로 커서가 이동하고 입력 모드 상태가 된다.

```
unix vi test
I like unix      → 명령 모드 상태에서 o를 입력한다.
  x
~
```

i 명령키와 a 명령키의 비교

입력 모드로 변환할 때 가장 많이 사용하는 명령키인 i와 a의 차이를 알아보자. “unix test”라는 문자열에 “vi”를 삽입해 “unix vi test”로 만들어 보자. 현재 커서가 “unix”의 “x”자 위에 있다고 가정한다.

```
unix test
  x
~
~
```

먼저, i 명령키로 입력 모드로 전환하고 “vi”(공백+vi)를 입력하면 다음과 같이 된다. 즉,

현재 커서가 있는 x의 앞에서부터 입력이 되어 결과적으로 “uni vix test”가 된다.

```
uni vix test      ← i 명령키를 입력한 후 "vi"를 입력한다.
~
~
```

반면, a 명령키로 입력 모드로 전환하고 “vi”를 입력하면 다음과 같이 커서가 있는 “x”의
다음부터 입력이 되어 “unix test”가 된다.

```
unix vi test      ← a 명령키를 입력한 후 "vi"를 입력한다.
~
~
```

즉, i 명령키는 커서 앞에 입력하고, a 명령키는 커서 뒤에 입력한다.

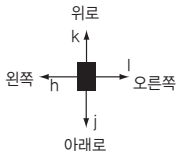
4 커서 및 화면 이동

편집기에서는 커서를 움직이며 작업하는 것이 기본이다. vi에서는 윈도우에서처럼 마우스로 커서를 이동시킬 수 없고 모두 키보드로 해야 한다. vi에서는 커서를 이동시키는 다양한 명령키를 제공한다.

커서 이동하기

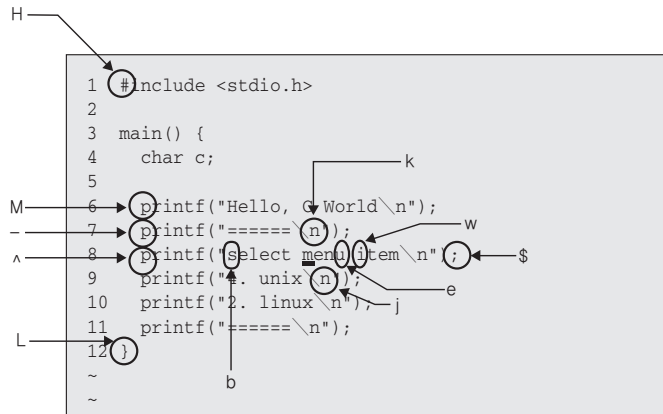
먼저 커서를 이동시키기 위해 많이 사용하는 명령키를 [표 4-4]에 정리하였다.

[표 4-4] vi 커서 이동 명령키

명령키	기능	
k	커서를 한 행 위로 이동시킨다.	
j	커서를 한 행 아래로 이동시킨다.	
l	커서를 한 문자 오른쪽으로 이동시킨다.	
h	커서를 한 문자 왼쪽으로 이동시킨다.	

^ 또는 0	커서를 현재 행의 처음으로 이동시킨다.
\$	커서를 현재 행의 마지막으로 이동시킨다.
-	커서를 이전 행의 처음으로 이동시킨다.
+ 또는 Enter	커서를 다음 행의 처음으로 이동시킨다.
H	커서를 화면의 맨 위 행으로 이동시킨다.
M	커서를 화면의 중간 행으로 이동시킨다.
L	커서를 화면의 맨 아래 행으로 이동시킨다.
w	커서를 다음 단어의 첫 글자 위치로 이동시킨다.
b	커서를 앞 단어의 첫 글자 위치로 이동시킨다.
e	커서를 다음 단어의 마지막 글자 위치로 이동시킨다.

[그림 4-4]를 참조해 커서 이동 명령키들의 동작을 살펴보자. [그림 4-4]에서 행 번호는 설명을 위한 표시로 내용과는 상관없다.



[그림 4-4] 커서 이동 명령키 예

우선 커서가 8행의 “menu”의 첫 글자 “m”에 있다고 가정한다. 커서를 좌우로 한 글자씩 움직이려면 l(소문자 L)이나 h 명령키를 사용한다. 윗 행으로 올라가려면 k 명령키를 사용한다. k 명령키를 사용하여 윗 행으로 이동하면 m의 바로 위인 n으로 이동하지만 - 명령키로 이동할 경우에는 7행의 첫 글자인 p로 커서가 이동한다. 현재 위치에서 w 명령키를 입력하면 다음 단어인 “item”의 첫 글자인 “i”로 커서가 이동한다. 다시 b 명령키를 입력하면 이전 단어인 “select”의 첫 글자인 “s”로 커서가 이동한다. e 명령키를 입력할 경우

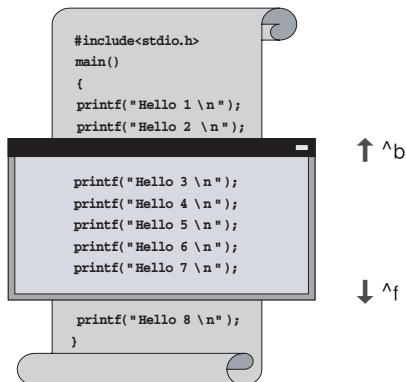
현재 단어인 “menu”의 마지막 글자인 “u”로 이동하고 다시 e 명령키를 입력하면 다음 단어인 “item”의 마지막 글자인 “m”으로 이동한다. 현재 행인 8행의 처음으로 이동하려면 ^ (캐럿) 명령키를 입력한다. 캐럿키는 [Shift]+6이다. 현재 행의 끝으로 이동하려면 \$ 명령키를 입력한다. 화면의 처음으로 이동하려면 H 명령키를, 중간으로 이동하려면 M 명령키를 화면의 끝(파일의 끝)으로 이동하려면 L 명령키를 입력한다.

5 화면 이동하기

vi의 기본 크기는 단말기의 화면 크기이다. 파일의 크기가 이보다 클 경우 화면을 이동해야 한다. vi에서는 윈도우에서처럼 [Page Up] / [Page Down] 키로 동작하지 않는다. 커서 이동 명령 키처럼 [표 4-5]와 같은 별도의 화면 이동 명령키들이 있다.

[표 4-5] vi 화면 이동 명령키

명령키	기능
^u	반 화면 위로 이동시킨다.
^d	반 화면 아래로 이동시킨다.
^b	한 화면 위로 이동시킨다.
^f	한 화면 아래로 이동시킨다.
^y	화면을 한 행만 위로 이동시킨다.
^e	화면을 한 행만 아래로 이동시킨다.



[그림 4-5] 화면 이동 명령키의 사용

[그림 4-5]와 같이 파일이 화면보다 클 경우 화면에는 일부만 보이게 된다. 이렇게 현재 보이는 화면을 이동시키려면 [표 4-5]에 나와 있는 명령키를 이용한다. **[Ctrl]+f** 명령키를 입력하면 현재 보이는 부분의 아래 부분이 나타나고, **[Ctrl]+b** 명령키를 입력하면 현재 보이는 부분의 위 부분이 나타난다.

지정한 행으로 이동하는 명령키

커서를 지정한 행으로 바로 이동시키기 위해서는 G(**[Shift]+g**) 명령키를 사용하거나 마지막 행 모드 명령을 사용한다. 명령 모드에서 그냥 G 명령키만 입력하면 파일의 마지막 행으로 이동하며 G 명령키 앞에 행 번호를 붙이면 해당 행으로 커서가 이동한다. 예를 들어 30G를 입력하면 커서가 30번째 행으로 바로 이동한다.

마지막 행 모드에서도 커서를 이동시킬 수 있다. :을 입력해 마지막 행으로 전환한 뒤 이동하려는 행 번호를 입력하면 해당 행으로 커서가 이동한다. 행 번호 대신 \$를 입력하면 파일의 마지막 행으로 이동한다. 예를 들어 :30을 입력하면 30번째 행으로 커서가 이동하는 것이다.

⑥ 수정 및 삭제

vi는 입력한 내용을 수정하거나 삭제하는 다양한 명령키도 다양하게 제공한다. 그리고 수정이나 삭제 동작을 취소하는 명령키도 있다.

내용 수정하기

내용의 수정 명령키에는 한 글자만 수정할 수 있는 명령키, 단어별로 수정할 수 있는 명령키, 수정할 글자 수를 지정해 수정할 수 있는 명령키 등이 있다. 그런데 한 글자를 수정할 수 있는 r 명령키 외의 명령키들은 자동으로 입력 모드로 전환되기 때문에 수정을 마치면 **[Esc]** 키를 입력해야 한다.

[표 4-6] vi 내용 수정 명령키

명령키	기능
r	커서가 위치한 문자를 다른 문자로 수정한다.
cw, #cw	커서의 위치부터 현재 단어의 끝까지 수정한다(#에는 수정할 단어의 수를 지정, 예를 들어 3cw는 커서 위치부터 3단어 수정).
s, #s	커서의 위치부터 [Esc] 키를 입력할 때까지 수정한다(#에는 수정할 문자의 수를 지정, 예를 들어 5s는 커서 위치부터 5글자를 수정).
cc	커서가 위치한 행의 내용을 모두 수정한다.
C	커서의 위치부터 행의 끝까지 수정한다.

아래 예를 살펴보자. 먼저 커서는中间的 “wditor”의 첫 글자인 “w” 위에 있다. 이 “w”를 e로 고치기 위해서는 r 명령키를 사용하는 것이 좋다. r 명령키를 입력하고 다시 e 를 입력하면 간단하게 수정된다. [Esc] 키를 입력할 필요도 없다.

```
unix wditor test      → r 명령키를 입력해 커서가 위치한 문자를 수정한다.
~
```

이제 “editor” 전체를 다른 단어로 바꾸어 보자. vi에서는 단어를 공백 문자나 특수 문자로 구별한다. 이 경우에는 공백 문자로 구별되어 “editor”를 한 단어로 인식한다. 이 때는 단어 수정 명령키인 cw나 s 명령키에 “editor”의 글자 수인 6을 덧붙여 6s를 이용할 수 있다. cw나 6s 명령키를 입력하면 끝 글자를 “\$”로 바꾸어 “e”부터 어디까지 수정할 것인지 표시해 준다.

```
unix edito$ test      → cw나 6s 명령키를 입력해 커서가 위치한 문자부터 $까지 수정한다.
~
```

현재 커서가 위치한 “e”부터 행의 끝까지 수정하려면 대문자 C 명령키를 이용한다. 그러면 행의 마지막 글자가 “\$”로 바뀌어 모두 수정할 것임을 표시한다. cc 명령키를 입력하면 현재 행의 모든 내용이 삭제되고 커서는 행의 처음으로 이동해 새로운 입력을 기다린다.

```
unix editor tes$ → C 명령키를 입력해 커서가 있는 곳부터 행의 끝까지 수정한다.  
~
```

내용 삭제하기/취소하기

입력 모드에서 입력 중에 틀린 글자를 삭제할 때는 ⏮ 키나 Delete 키를 이용하면 된다. 명령 모드에서는 문자를 삭제하거나 행을 삭제하기 위해 별도로 지정된 명령키들이 있다. 삭제하려는 문자 수나 행 수를 지정하려면 명령키 앞에 숫자를 지정하면 된다. 예를 들어 한 글자를 삭제하는 것은 x 명령키이지만 5글자를 삭제하려면 5x를 입력하면 된다.

[표 4-7] vi 내용 삭제 및 취소 명령키

명령키	기능
x, #x	커서가 위치한 문자를 삭제한다(#은 삭제할 문자 수, 예를 들어 3x하면 3글자 삭제).
dw, #dw	커서 위치의 단어를 삭제한다(#은 삭제할 단어 수).
dd, #dd	커서가 위치한 행을 삭제한다(#은 삭제할 행의 수, 예를 들어 5dd는 커서 위치부터 5행을 삭제).
⌘ + d(D)	커서 위치부터 행의 끝까지 삭제한다.
u	방금 수행한 명령을 취소한다.
U	해당 행에서 한 모든 명령을 취소한다.
:e!	마지막으로 저장한 내용 이후의 것을 버리고 새로 작업한다.

다음 예에서 현재 커서가 위치한 “e”라는 글자를 지우려면 x 명령키를 입력하면 된다. “editor”를 모두 지우려면 6x나 dw 명령키를 이용한다.

```
unix editor test  
~
```

현재 행을 지우려면 dd 명령키를 입력하고 현재 위치부터 행의 끝까지 모두 지우려면 대문자 D 명령키를 입력한다.

```
unix → D 명령키를 입력하면 커서가 있는 곳부터 행의 끝까지 모두 지운다.  
~
```

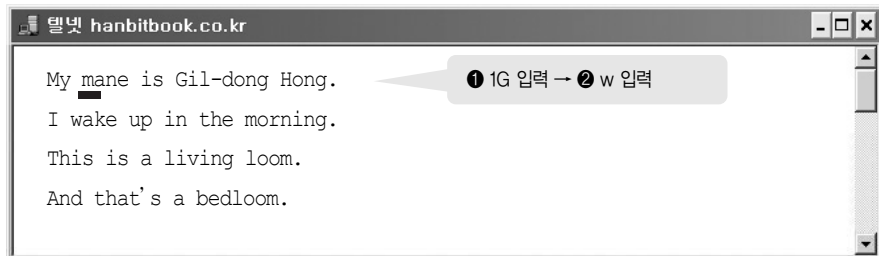

:w : 작업한 내용을 저장한다.



3 커서 이동하기 : 커서를 1행의 두 번째 단어인 mane로 이동시킨다.

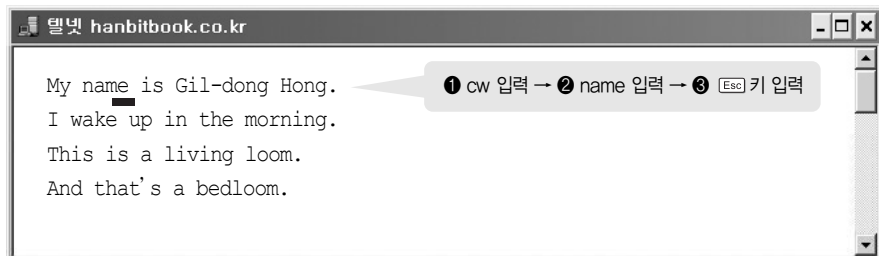
G : 커서를 지정한 행으로 옮긴다. 여기서는 1G이므로 1행으로 옮긴다.

w : 커서를 다음 단어의 첫 글자, 즉 mane의 m으로 옮긴다.

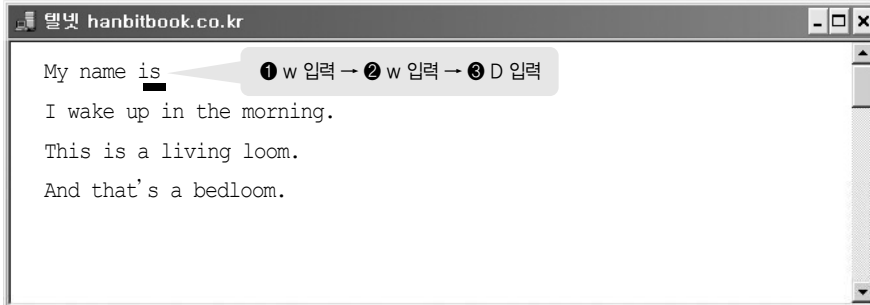


4 단어 수정하기 : 1행의 두 번째 단어인 mane를 name로 수정한다.

cw : 현재 커서가 있는 위치부터 단어 끝까지 수정하므로 여기서는 mane가 된다. cw 명령키는 자동으로 입력 모드로 전환하므로 내용 입력 후 [Esc]키를 눌러 명령 모드로 전환한다.



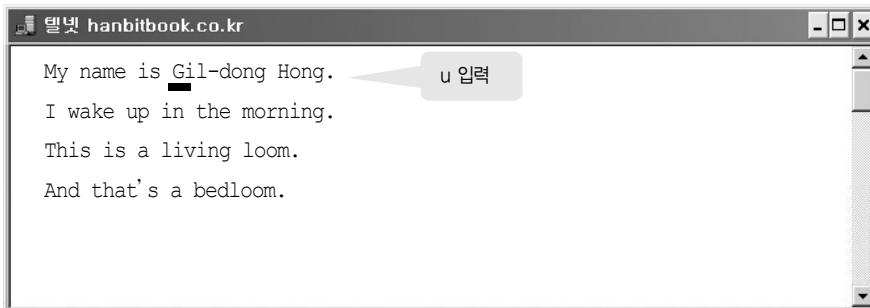
- 5 이름을 삭제하기 : 1행의 is 다음에 있는 이름을 모두 삭제한다.



w : 커서를 다음 단어의 첫 글자로 옮기므로 여기서는 w를 두번 입력해 커서를 Gil-dong의 G로 옮긴다.

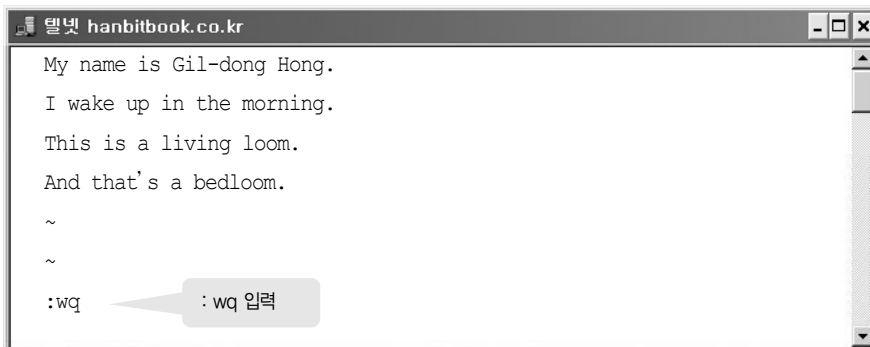
D : 커서가 있는 위치부터 행의 끝까지 모두 삭제한다.

- 6 이전 동작 취소하고 복구하기 : 1행에서 삭제한 Gil-dong Hong의 삭제를 취소하고 복구한다.



u : 이전 명령을 취소하므로 여기서는 바로 전에 삭제한 내용을 복구한다.

- 7 저장하고 종료하기 : 작업 내용을 저장하고 종료한다.



:wq 작업한 내용을 저장하고 vi를 종료한다.

8 디렉토리 위치 변경하기 : 실습 디렉토리에서 빠져나간다.



```
$ cd ..  
$ pwd  
/export/home/user/ch4  
$
```

7 vi 편집 기능

vi는 커서의 이동과 수정, 삭제 외에 복사와 붙이기, 잘라내기, 블록 지정, 검색과 치환 등 일반적인 편집기에서 제공하는 다양한 기능을 제공한다.

복사하기/붙이기/잘라내기 명령

vi에서는 복사하기와 붙이기도 마우스 대신 명령키를 사용한다. 소문자 yy 명령키를 입력하면 해당 행이 복사되고, 원하는 위치로 커서를 이동해 p 명령키를 입력하면 붙이기가 된다. 앞서 배운 dd 명령키로 행을 삭제한 후 p 명령키를 입력하면 “잘라내서 붙이기”가 된다. 여러 행을 복사하려면 5yy 명령키처럼 숫자를 붙이면 된다. 5yy 명령키는 현재 커서가 있는 행부터 5행을 복사한다. 복사하거나 잘라두기를 한 뒤에는 다른 명령을 사용하지 말고 원하는 위치로 바로 이동해 붙이기를 하는 것이 좋다. 다른 명령을 사용하다 임시 버퍼에 저장된 내용을 잃어버릴 수 있기 때문이다.

[표 4-8] vi 복사하기와 붙이기 명령키

명령키	기능
yy, #yy	커서가 위치한 행을 복사한다(#에는 복사할 행의 수 지정, 예를 들어 3yy하면 3행 복사)
p	커서가 위치한 행의 아래쪽에 붙인다.
P	커서의 위치한 행의 위쪽에 붙인다.
dd, #dd	커서가 위치한 행을 잘라둔다(#에는 잘라줄 행의 수 지정, 예를 들어 3dd 명령을 입력하면 3행 잘라두기(삭제)).

다음 예를 이용해 복사하기와 붙이기 명령키의 동작을 살펴보자. 이번에도 행 번호는 편의를 위해 표시한 것이다.

```
1      Hello, Unix World!!
2      =====
```

커서가 1행에 있을 경우 1행만 복사하려면 그냥 yy 명령키만 입력하면 된다. 1행과 2행을 함께 복사할 경우에는 2yy 명령키를 입력한다. 그리고 원하는 위치로 이동해 붙이기(p 명령키)를 하면 된다. 1행에서 2yy 명령키를 입력해 2행에서 붙이기를 하였다면 결과는 다음과 같다.

```
1      Hello, Unix World!!
2      =====
3      Hello, Unix World!!
4      =====
```

dd 명령키는 삭제뿐만 아니라 잘라서 붙이기를 할 때도 사용된다. 위 예에서 2행을 잘라 1행 위에 붙이기를 한다고 하면 커서를 2행으로 이동하고 dd 명령키를 입력한 다음 커서를 1행으로 이동해 대문자 P 명령키를 입력하면 된다. 결과는 다음과 같다.

```
1      =====
2      Hello, Unix World!!
3      Hello, Unix World!!
4      =====
```

버퍼의 사용

버퍼는 복사하거나 잘라낸 내용을 저장해두는 임시 저장 공간으로 윈도우의 클립보드와 같은 기능을 수행한다. yy 명령키를 사용해 복사할 경우 복사된 내용은 버퍼에 저장된다. 이렇게 이름을 붙이지 않은 버퍼를 언네임드 버퍼라고 한다. 이와 달리 버퍼에 이름을 붙여서 사용할 수 있는데 이를 네임드 버퍼라고 한다. 언네임드 버퍼는 한번에 하나의 내용 밖에 저장할 수 없다. 예를 들어, 2yy 명령키로 두 행을 복사해서 저장했는데 다시 3yy 명령키

를 입력하면 앞에 저장한 두 행은 잃어버리게 되는 것이다. 이런 경우 네임드 버퍼를 사용하면 각기 다른 이름을 붙인 버퍼에 별도로 저장할 수 있어 원하는 내용을 모두 저장하고 사용할 수 있다.

네임드 버퍼에서 버퍼의 이름을 붙일 때는 다음과 같이 “+문자 또는 “+숫자의 형태로 사용한다. 숫자를 사용할 경우 숫자 버퍼라고도 한다.

- 네임드 버퍼 : “a, “b, “c, “d, ..., “z
- 숫자 버퍼 : “1, “2, ..., “9

네임드 버퍼에 커서가 위치한 행을 저장하려면 “ayy하면 된다. 이를 붙이기 하려면 “ap하면 된다.

```
1      Hello, Unix World!!
2      =====
```

다시 앞의 예를 살펴보자. 커서가 1행에 있을 때 2yy 명령키를 입력하면 1행, 2행 순서대로 언네임드 버퍼에 저장되어 있어 붙이기를 하면 다시 1행, 2행 순서대로 붙여진다. 만약 1행과 2행을 따로 복사해 붙이기를 하려고 한다면 네임드 버퍼를 사용해야 한다. 즉, 1행에서 “a로 a 버퍼를 지정하고 yy 명령키로 복사한 뒤 2행으로 커서를 이동해 “b 명령키로 b 버퍼에 2행을 복사하면 된다. 이렇게 각각 복사한 것을 2행 다음에 b 버퍼 먼저, a 버퍼 나중 순서로 붙이기를 하려면 “bp→”ap 순서대로 명령키를 입력하면 된다. 결과는 다음과 같다.

```
1      Hello, Unix World!!
2      =====
3      =====
4      Hello, Unix World!!
```

마지막 행 모드에서 복사하기와 잘라내기

마지막 행 모드에서도 행을 복사하고 잘라내기를 할 수 있다. 행을 복사하거나 잘라낼 때는 범위를 지정해서 할 수 있다.

■ 범위 지정하기

윈도우에서 마우스로 드래그해 범위를 지정하는 것을 vi에서는 명령으로 지정한다. 범위 지정에는 숫자와 특수 문자를 사용한다. 범위를 지정하기 위해서는 우선 :을 입력해 마지막 행 모드로 이동해야 한다. 범위 지정에서 .은 커서가 위치한 현재 행을 나타내고, \$는 마지막 행을 의미한다.

[표 4-9] vi 범위 지정 명령키

명령키	의미
1, \$ 또는 %	1행부터 마지막 행까지 지정한다.
1,.	1행부터 커서가 있는 행까지 지정한다.
.,\$	커서가 있는 행부터 마지막 행까지 지정한다.
.-3	현재 행부터 앞으로 세 번째 행까지 지정한다.
10,20	10행부터 20행까지 지정한다.

■ 복사하기 / 잘라내기

마지막 행 모드에서 사용하는 복사와 잘라내기 명령들을 [표 4-10]에 정리하였다. 잘라 내기는 삭제와 동일한 기능을 수행한다. 명령의 앞에 붙은 :은 마지막 행 모드로 이동하기 위한 것이다. 이미 마지막 행 모드에 있다면 :을 제외하고 명령을 입력하면 된다.

[표 4-10] vi 마지막의 행 모드에서의 이동/복사/삭제 명령키

명령키	기능
:#y	#으로 지정한 행을 복사한다(예: 10y → 10행을 복사).
:<범위>y	범위로 지정한 행을 복사한다(예: 10,20y → 10행~20행까지 복사).
:#d	#으로 지정한 행을 삭제한다(예: 10d → 10행 삭제).
:<범위>d	범위로 지정한 행을 삭제한다(예: 10, 20d → 10행~20행까지 삭제).
:pu	현재 행 다음에 버퍼의 내용을 붙인다.
:#pu	#으로 지정한 행 다음에 버퍼의 내용을 붙인다(예: 5pu).

범위 지정과 함께 마지막 행 모드에서 사용하는 명령들을 살펴보자. 다음 예에서 커서는 2행에 있다. 이 때 커서를 3행으로 보내려면 :3을 입력하면 된다. 현재 예에서는 3행이

마지막 행이므로 :\$ 명령을 사용해도 된다. :2y 명령은 2행을 복사하라는 명령이고, :2,3y는 범위 지정을 사용해 2행과 3행을 복사하라는 명령이 된다. :1,\$d는 1행부터 마지막 행까지 모두 삭제하라는 명령이다.

```
1      Hello, Unix World!!
2      Unix Vi Editor Test
3      I like Unix.
~
```

:2y를 한 뒤에 :3pu를 하게 되면 다음과 같이 3행 다음에 2행을 붙인 결과가 나온다.

```
1      Hello, Unix World!!
2      Unix Vi Editor Test
3.     I like Unix.
4      Unix Vi Editor Test
~
```

검색하기/바꾸기

vi에서 특정 단어를 검색하거나 검색한 단어를 다른 단어로 바꾸기 위한 명령은 마지막 행 모드에서 제공한다.

■ 검색하기

검색을 위해 마지막 행 모드로 가려면 /나 ?를 먼저 입력해야 한다. /나 ?를 입력하면 커서가 마지막 행으로 이동한다. /는 커서의 위치에서 아래 방향으로 검색을 시작하고, ?는 커서의 위치에서 위 방향으로 검색을 한다. /나 ? 다음에 찾고자 하는 문자열을 입력하고 **[Enter]** 키를 누르면 검색을 하고, 문자열을 찾으면 해당 문자열의 시작으로 커서가 이동한다. 이때 다음 발견 문자열로 이동하려면 n(next) 명령을 사용한다.

[표 4-11] vi 검색 명령키

명령키	기능
/문자열	문자열을 아래 방향으로 검색한다.
?문자열	문자열을 위 방향으로 검색한다.
n	원래 찾는 방향으로 다음 문자열을 찾는다.
N	역방향으로 다음 문자열을 찾는다.

다음 예에서 “Unix”를 검색하기 위해 /를 입력하면 커서가 마지막 행으로 이동하고, 찾고자 하는 “Unix”를 입력하고 [Enter] 키를 누르면 현재 커서 위치가 2행의 “Unix”이므로 커서가 그대로 있다.

```

1      Hello, Unix World!!
2      Unix Vi Editor Test
3      I like Unix.
/Unix

```

다시 다음 “Unix”를 검색하려고 n을 입력하면 3행의 “Unix”로 커서가 이동한다. 여기서 다시 n을 입력하면 아래와 같이 “찾기가 버퍼의 끝주변을 지남”이란 메시지가 나오면서 다시 1행의 “Unix”로 커서가 이동한다.

```

1      Hello, Unix World!!
2      Unix Vi Editor Test
3      I like Unix.
찾기가 버퍼의 끝주변을 지남

```

■ 바꾸기

기존의 문자열을 다른 문자열로 바꾸기 위해서는 먼저 :를 입력해서 마지막 행 모드로 이동한다. 바꾸기 명령은 파일 전체를 대상으로 할 수도 있고 특정 범위만 지정해 실행할 수도 있다.

[표 4-12] vi 바꾸기 명령키

명령키	기능
:s/문자열1/문자열2/	커서가 위치한 행의 첫번째 문자열1을 문자열2로 바꾼다.
:%s/문자열1/문자열2/g	파일 전체에서 모든 문자열1을 문자열2로 바꾼다.
:<범위>s/문자열1/문자열2/	<범위> 안의 모든 행에 대해서 각 행의 첫 번째 문자열1을 찾아 문자열2로 바꾼다.
:<범위>s/문자열1/문자열2/g	<범위> 안의 모든 행에서 문자열1을 문자열2로 바꾼다.
:<범위>s/문자열1/문자열2/gc	<범위> 안의 모든 행에 대해서 각 문자열1을 문자열2로 바꿀 때 수정할지 여부를 묻는다

다음 예에서 “Unix”를 “Linux”로 바꾸는 동작을 [표 4-12]의 명령 차례대로 살펴 보자.

```

1      Hello, Unix World!!
2      Unix Vi Editor Test, Unix
3      I like Unix.
```

커서는 2행의 첫 번째 “Unix”위에 있다고 가정한다. 우선 “:s/문자열1/문자열2/”을 보면 명령은 “:s/Unix/Linux/”가 된다. 2행에는 “Unix”가 두 개 있는데 이 명령은 그 중 맨 앞에 있는 것만 바꾸어 준다. 따라서 결과는 “Linux Vi Editor Test, Unix”가 된다. 만약 2행에 있는 “Unix” 전체를 “Linux”로 바꾸고 싶으면 “:s/Unix/Linux/g”를 하면 된다.

다음으로 “:%s/문자열1/문자열2/g”를 적용하면 “:%s/Unix/Linux/g”가 된다. 범위 지정에서 “%”는 전체 행을 뜻한다. “%” 대신에 “1,\$s/Unix/Linux/g”를 해도 된다. 따라서 이 명령은 파일 내의 모든 “Unix”를 “Linux”로 바꾸라는 것이 된다. 이 명령에서 마지막의 “g”를 지정하지 않으면 2행의 두 번째 “Unix”는 바뀌지 않는다. 파일의 일부 행만 바꾸고 싶을 경우 “:1,2s/Unix/Linux/g”와 같이 하면 된다. 이는 1행과 2행에서 “Unix”를 “Linux”로 바꾸라는 명령이다. 마지막 명령 형태인 “:<범위>s/문자열1/문자열2/gc”를 적용해 보자. 적용 형태는 “:2,3s/Unix/Linux/gc”가 된다. c는 사용자에게 문자열을 바꿀 것인지 확인하게 한다. 이 때 y 명령키와 Enter 키를 입력해야 문자열이 바뀐다.


```

1      Hello, Unix World!!
2      Unix Vi Editor Test, Unix
3.     I like Unix.
~
~
:2,3s/Unix/Linux/gc
Unix Vi Editor Test, Unix
^^^^y
Linux Vi Editor Test, Unix
          ^^^^

```

실습하기 vi 편집 방법 익히기

- 1 실습용 디렉토리 만들기 :** 홈 디렉토리 아래에 Unix/ch4/Practice 디렉토리를 생성한다.

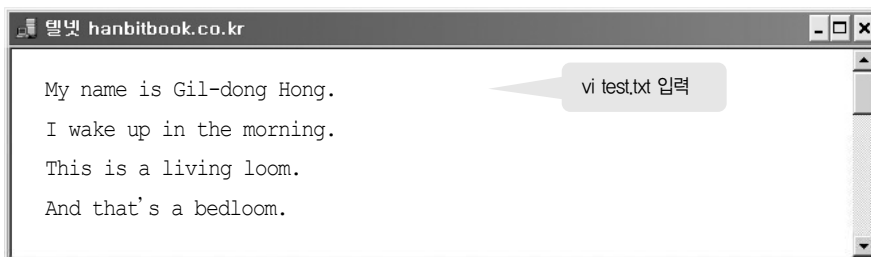


```

$ pwd
/export/home/user1/Unix/ch4
$ mkdir Practice
$

```

- 2 실습용 예제 파일 열기 :** vi test.txt 명령으로 예제 파일을 연다.



```

My name is Gil-dong Hong.
I wake up in the morning.
This is a living loom.
And that's a bedroom.

```

vi test.txt 입력

G : 커서를 지정한 행으로 옮긴다. 여기서는 1G이므로 1행으로 옮긴다.

yy : 커서가 있는 행을 복사하므로 여기서는 1행을 복사한다.

j : 커서를 한 행 아래로 옮긴다.

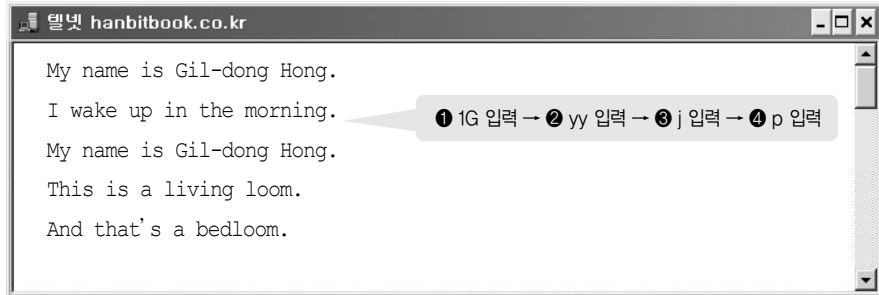
p : 커서가 있는 행의 아래 행에 붙인다.

/문자열 : 문자열을 아래 방향으로 검색한다.

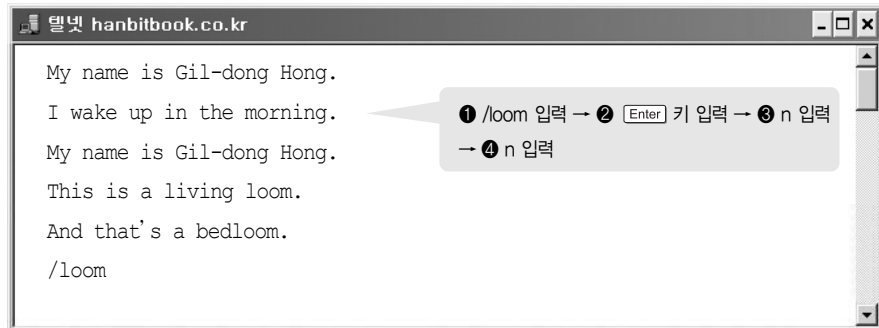
n : 원래 찾는 방향으로, 여기서는 아래 방향으로 다음 문자열을 찾는다.

<범위>문자열1/문자열2/g는 <범위> 안의 모든 행에 대하여 문자열1을 문자열2로 바꾼다. <범위>가 %인 것은 모든 행을 의미하므로 여기서는 파일 전체에서 loom을 room으로 바꾼다.

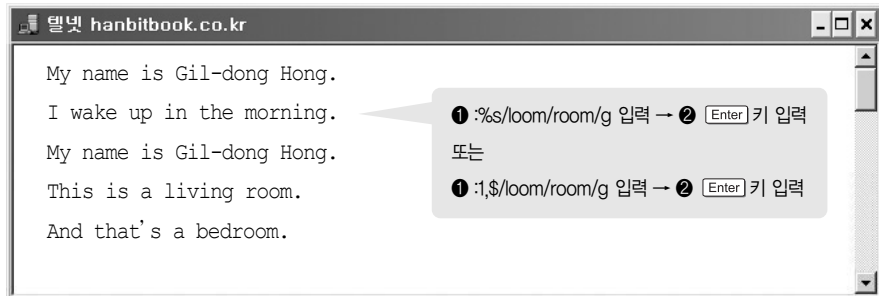
3 복사해 붙이기 : 1행을 복사해 3행 다음에 붙인다.



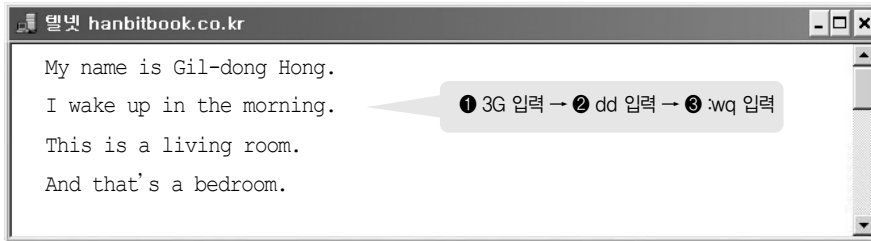
4 단어 검색하기 : 파일에서 loom이란 단어를 검색한다.



5 단어 바꾸기 : 파일에서 loom이란 단어를 모두 찾아 room으로 바꾼다.



6 행 삭제한 후 종료한 후 종료하기 : 3행을 삭제한 뒤 저장하고 종료한다.



G : 커서를 지정한 행으로 옮긴다. 여기서 3G이므로 3행으로 옮긴다.

dd : 커서가 위치한 행을 삭제하므로 여기서 3행을 삭제한다.

:wq : 작업한 내용을 저장하고 vi를 종료한다.

7 디렉토리 위치 변경하기 : 실습 디렉토리에서 빠져나간다.



8 기타 유용한 명령키

지금까지 살펴본 주요 명령 외에 파일을 읽어오거나, vi 작업 도중 셸 기능 수행, 이전 명령의 반복 수행, 화면 다시 출력 등 vi에서 유용하게 사용할 수 있는 명령들이 있다.

파일 읽어오기/여러 파일 편집하기 명령키

vi에서 다른 파일을 읽어들여 작업을 하거나 작업 중이던 파일을 종료하고 다른 파일로 작업을 전환하기 위한 명령들도 마지막 행 모드에서 실행한다. vi를 시작할 때 여러 개의 파일명을 지정했을 경우 다음 파일로 이동할 때는 :n 명령을 사용한다.

[표 4-13] vi 파일 편집 명령키

명령키	기능
:r 파일명	지정한 파일을 읽어 현재 커서 위치로 삽입한다.
:e 파일명	지정한 파일로 전환한다(기존 파일을 :w로 저장한 뒤에 실행해야 함).
:n	vi 시작시 여러 파일을 지정하였을 경우 다음 파일로 작업을 이동한다.

■ 파일 읽어들이기

다른 파일 읽어들이는 예를 살펴보자. 먼저 예제 파일로 다음 두 개를 사용한다.

파일 1 : first	파일 2 : second
1 Hello, Unix World!!	Hi, C World.
2 Unix Vi Editor Test, Unix	C Test.
3 I like Unix.	I like C.

현재 작업 중인 파일이 first이고 커서가 2행에 있을 때 :r second를 실행하면 second 파일의 내용이 first파일의 2행 다음에 삽입된다. 결과를 살펴보면 다음과 같다.

파일 1 : first	파일 2 : second
1 Hello, Unix World!!	Hi, C World.
2 Unix Vi Editor Test, Unix	C Test.
3 Hi, C World.	I like C.
4 C Test.	
5 I like C.	
6 I like Unix.	

■ 다른 파일 편집하기

:e는 현재 작업 중인 파일을 종료하고 다른 파일을 편집하려고 할 때 사용하면 편리하다. 예를 들어 first 파일을 편집하다가 second 파일 편집으로 바꾸려면 :e second하면 된다. 이 때 작업 중인 파일을 먼저 저장하고 :e 명령을 실행해야 한다. 파일을 저장하지 않고 그냥 :e second을 실행하면 다음과 같은 오류 메시지가 출력된다.

```
변경된 부분이 보관되지 않았음 (:edit! 무시)
```

vi를 종료하지 않고 저장만 하려면 :w만 하면 된다. 작업 중인 내용을 저장하지 않고 다른 파일 편집으로 가고 싶을 경우에는 :e! second를 해야 한다.

■ 여러 파일 편집하기

vi를 시작할 때 파일명을 여러 개 지정할 수 있다. 예를 들어 다음과 같이 할 수 있다.

```
$ vi first second
```

파일 first와 second 두 개 모두 작업하겠다는 뜻이다. 이렇게 지정하면 앞에 지정한 파일이 먼저 열리고, 다시 다음 파일로 이동하려고 할 때 :n을 해야 한다. 물론 파일을 수정했으면 저장 후 :n을 해야 한다. :n으로 파일 second로 이동한 다음에는 :n 명령으로 다시 first로 이동할 수 없다. 그냥 :e first 명령을 이용해야 한다.

vi에서 셸 명령 사용하기

vi 작업 도중 작업 디렉토리의 파일 목록을 확인하거나, 프로그램을 컴파일하는 등 셸 명령을 실행해야 할 경우가 있다. 이 때 vi를 종료하지 않고 셸 명령을 실행할 수 있다.

[표 4-14] vi에서 셸 명령 사용 방법

명령	기능
:! 셸명령	vi 작업을 잠시 중단하고 셸 명령을 실행한다(vi로 돌아오려면 <code>[Enter]</code> 키 입력).
:sh	vi를 잠시 빠져 나가서 셸 명령을 실행한다(vi로 돌아오려면 <code>exit</code> 명령 입력).

■ :! 기능 이용하기

vi 작업 도중 셸 명령을 실행하는 가장 간단한 방법이다. “:! 셸 명령”과 같은 형태로 한다. 예를 들어 :! date하면 아래와 같은 형태로 실행된다.

```
~
:! date
2006년 1월 20일 금요일 오후 09시 03분 31초
[계속하려면 리턴 키를 치십시오]
```

vi를 빠져나가거나 하는 번거로움 없이 바로 이용할 수 있다는 장점이 있다. 셸 명령의 결과를 확인하고 다시 vi 작업으로 돌아가려면 `[Enter]` 키만 입력하면 된다.

■ :sh 기능 이용하기

“:! 셸 명령”은 실행할 셸 명령이 여러 개일 경우 매번 :! 셸 명령을 실행해야 하므로 불편할 수 있다. 이런 경우에는 vi를 잠시 빠져나가 작업을 수행하고 다시 돌아오는 것이 편리하다. 이 때 :sh을 하면 vi를 잠시 빠져나갈 수 있다. 셸 명령 작업을 마치고 다시 vi로 돌아오려면 exit를 입력하면 된다.

```
~
:sh
$ date
2006년 1월 20일 금요일 오후 09시 07분 39초
$ exit
```

기타 명령 알아보기

이 밖에 vi를 편리하게 사용하기 위해 알아두어야 할 명령키들은 [표 4-15]와 같다.

[표 4-15] vi 기타 명령키

명령	기능
^(소문자 l)	화면을 다시 출력한다.
^g	현재 행을 마지막 행에 출력한다.
[Shift] + j(대문자 J)	현재 행과 아래 행을 연결하여 하나의 행으로 만든다.
.	바로 직전에 했던 명령을 반복한다.

■ 화면 다시 출력하기

vi 작업 도중에 시스템 메시지가 출력된다든지 다른 사용자가 메시지를 보내서 화면이 정상이 아닐 때 [Ctrl]+l 명령키를 입력하면 메시지들이 사라지고 원래 작업 중이던 내용만 남는다.

■ 행 연결하기

행을 연결할 때는 J 명령키를 사용한다. J 명령키는 커서가 위치한 행과 다음 행을 하나의 행으로 만들어준다. 다음 예에서 현재 커서가 위치한 첫 번째 행에서 J 명령키를 입력해 보자.

```
unix
  █
editor
test
~
```

두 번째 행과 합쳐진다. 커서는 새로 붙여진 단어의 첫 글자로 이동한다.

```
unix editor
test
  █
~
```

여기서 다시 J 명령키를 입력하면 다시 또 아래 행을 위 행과 붙여준다.

```
unix editor test
  █
~
```

■ 이전 명령 반복하기

.(점) 명령키는 반복이다. 바로 앞에 했던 명령을 반복적으로 수행해 준다. 위에서 두 번째 J 명령키 대신에 .을 입력해도 같은 결과가 나온다. 바로 이전에 했던 명령이 J였기 때문이다. 단, 커서 이동키는 . 명령키를 입력해도 반복하지 않는다.

■ 대소문자 전환하기

~(틸드) 명령키는 커서가 위치한 글자를 소문자는 대문자로, 대문자는 소문자로 바꿔준다. 위 예에서 커서가 t 위에 있을 때 ~ 키를 입력하면 T로 바뀌고 커서는 한 글자 앞으로 이동한다.

```
unix editor Test
  █
~
```

실습하기 기타 유용한 명령 익히기

- 1 실습용 디렉토리 만들기 :** 홈 디렉토리 아래에 Unix/ch4/Practice 디렉토리를 생성한다.

```

$ pwd
/export/home/user1/Unix/ch4
$ mkdir Practice
$

```

- 2 실습용 예제 파일 하나 더 만들기 :** 실습용 예제 파일을 하나 더 만든다. 파일명은 test2.txt로 한다

```

My major is
Computer
Science.
~
~
~
:wq

```

① vi test2.txt 입력 → ② i 입력 → ③ 내용 입력 → ④ :wq 입력

- 3 다른 파일 읽어오기 :** test.txt 파일을 열고, 마지막 행 다음에 test2.txt 파일을 읽어들인다

```

My name is Gil-dong Hong.
I wake up in the morning.
This is a living room.
And that's a bedroom.
My major is
Computer
Science.

```

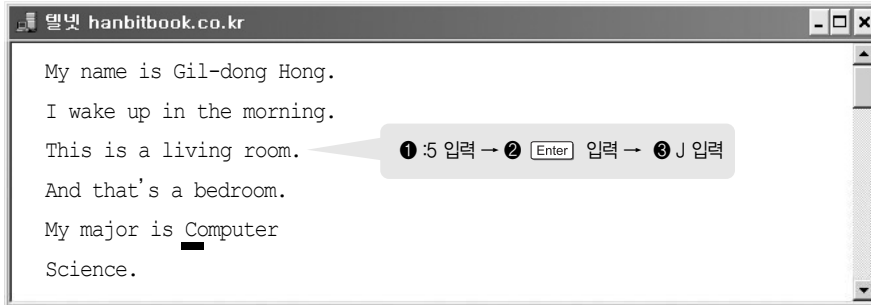
① vi test.txt 입력 → ② G 입력 →
③ :r test2.txt 입력 → ④ [Enter] 입력

i : 커서 앞에 입력한다.

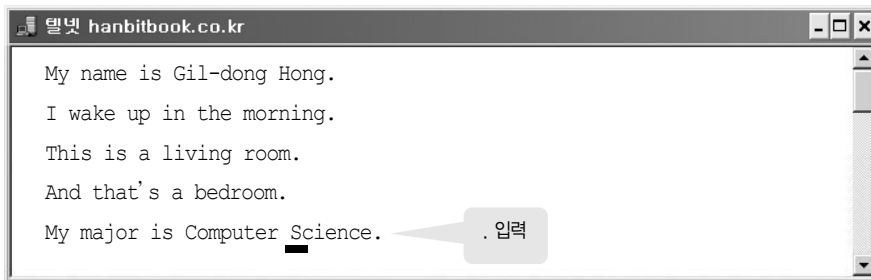
:wq 작업한 내용을 저장하고 vi를 종료한다.

G : 커서를 지정한 행으로 옮긴다. G만 입력하면 마지막 행으로 이동한다.

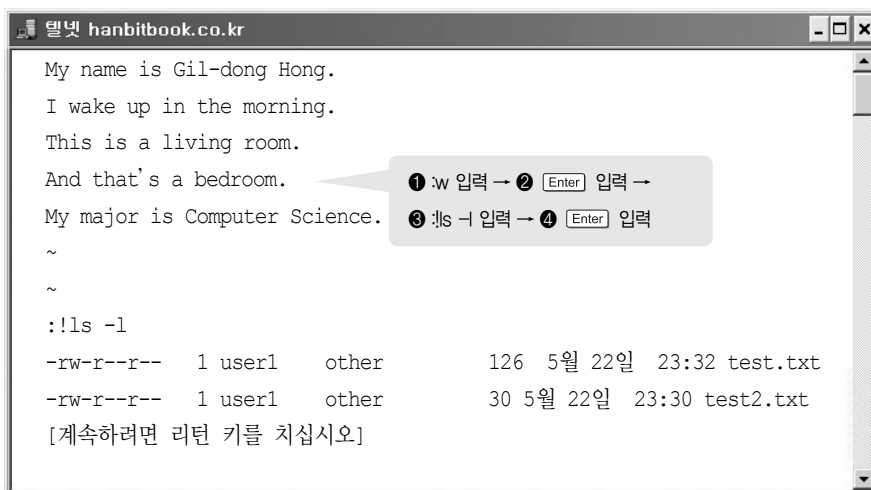
r 파일명 : 지정한 파일을 열어 현재 커서 위치로 삽입한다.

4 행 연결하기 : 5행과 6행을 연결해 한 행으로 만든다

J : 커서가 위치한 행과 다음 행을 붙여준다.

5 이전 동작 반복하기 : 이전 동작 반복 명령키를 사용해 5행과 6행을 한 행으로 연결한다

. : 이전 명령을 반복한다.

6 저장하고 셸 기능 사용하기 : test.txt 파일을 저장하고, 셸 명령을 실행시켜 파일의 크기를 확인한다

:w : 작업한 내용을 저장한다.

7 디렉토리 위치 변경하기 : 실습 디렉토리에서 빠져나간다.

A terminal window titled 'hanbitbook.co.kr' with standard window controls. It displays a sequence of shell commands and their output. The commands are: '\$ cd ..', '\$ pwd', and '\$'. The output for the 'pwd' command is '/export/home/user/ch4'.

```
$ cd ..  
$ pwd  
  /export/home/user/ch4  
$
```



vi의 환경설정

vi를 사용할 때 편리하도록 환경을 설정할 수 있다. vi의 환경설정도 마지막 행 모드에서 실행한다. 자주 사용되는 환경설정 방법을 살펴보면 [표 4-16]과 같다.

[표 4-16] vi 환경설정 명령키

명령키	기능
:set nu	파일 내용의 각 행에 행 번호 표시한다(보이기만 할 뿐 저장은 되지 않는다).
:set nonu	행 번호를 감춘다.
:set list	눈에 보이지 않는 특수 문자 표시한다(tab:^\, eol:\$ 등).
:set nolist	특수 문자를 감춘다.
:set showmode	현재 모드를 표시한다.
:set noshowmode	현재 모드 표시를 감춘다.
:set	set으로 설정한 모든 vi 변수를 출력한다.
:set all	모든 vi 변수와 현재 값을 출력한다.

행 번호 표시하기

:set nu를 하면 다음 예와 같이 행 번호가 표시된다. 예를 보자.

파일 : first	set nu 실행 후
Hello, Unix World!!	1 Hello, Unix World!!
Unix Vi Editor Test, Unix	2 Unix Vi Editor Test, Unix
I like Unix.	3 I like Unix.
:set nu	

행 번호는 사용자의 편의를 위해 보이는 것으로 파일에 저장되지는 않는다. :set nonu를 하면 행 번호는 다시 없어진다.

특수 문자 표시하기

행의 끝이나 탭 같은 특수 문자들은 vi에서 보이지 않는다. 이를 보려면 :set list 명령을 입력해야 한다. 아래 한글의 조판 부호 보기 기능과 비슷하다고 생각하면 된다.

파일 : first	set list 실행 후
Hello, Unix World!!	1 Hello, Unix World!! \$
Unix Vi Editor Test, Unix	2 Unix Vi ^I Editor Test, Unix \$
I like Unix.	3 I like Unix. \$
:set list	

first 파일에서 :set list를 하면 특수 문자 \$와 ^I이 보인다. \$는 행의 끝을 표시하고 ^I(**Ctrl**+대문자 i)는 탭을 나타낸다.

현재 모드 표시하기

vi 작업 도중 지금이 명령 모드인지 삽입 모드인지 헷갈릴 때가 있다. 이럴 때를 대비해 현재 모드를 표시하도록 해주는 명령이 :set showmode이다. 이 명령을 실행하면 명령 모드인 경우 아무 표시도 없지만 삽입 모드인 경우 화면 하단에 삽입 모드라고 표시된다.

~	
:set showmode	삽입 모드

환경설정값 표시하기

:set 명령은 현재 사용 중인 vi의 환경설정 상태를 보여준다. :set 명령의 결과는 다음과 같다. 시스템이 기본값으로 설정한 것과 사용자가 :set xxx 명령으로 설정한 것이 표시된다.

~
list redraw showmode term=ansi

모든 환경변수 표시하기

vi의 환경설정에 사용되는 모든 환경변수와 현재 설정값을 보여주는 명령은 `:set all`이다. 이 명령의 실행 결과는 다음과 같다.

```
~
:set all
noautoindent      nomodelines      showmode
autoprint          nonumber          noslowopen
noautowrite        nonovice          tabstop=8
nobeautify         nooptimize        taglength=0
directory=/var/tmp paragraphs=IPLPPPQPP LIpplpipnpb tags=tags /usr/lib/tags
noedcompatible     prompt            tagstack
noerrorbells       noreadonly        term=ansi
noexrc             redraw           noterse
flash             remap             timeout
hardtabs=8         report=5           ttytype=ansi
noignorecase       scroll=13           warn
nolisp             sections=NHSHH HUuhsh+c window=26
list              shell=/bin/ksh     wrapscan
magic             shiftwidth=8       wrapmargin=0
mesg              noshowmatch        nowriteany

[계속하려면 리턴 키를 치십시오]
```



요약

1 vi의 동작 모드

vi는 명령 모드와 입력 모드, 마지막 행 모드로 구분된다. 명령 모드에서 입력한 키들은 커서 이동이나 화면 이동 같은 명령으로 해석되어 실행된다. 입력 모드에서 입력한 키들은 보통의 편집기처럼 내용으로 입력된다. 마지막 행 모드는 화면의 가장 하단으로 커서가 이동해 특별한 명령들을 사용할 수 있는 모드이다.

2 vi의 명령

기능	명령 모드	마지막 행 모드
입력 모드로 이동	a, i, o, A, I, O	
마지막 행 모드로 이동	;, /, ?	
커서 이동	h, j, k, l, H, M, L, \$, ^, -, +, <input type="button" value="Enter"/> , w, b, e	
화면 이동	^f, ^b, ^u, ^d, ^y, ^e	
저장 및 종료	ZZ	:q, :q!, :w 파일명, :wq, :wq!,
내용 수정	r, cw(#cw), s(#s), cc, C	
내용 삭제	x(#x), dw(#dw), dd(#dd), D	:#d, :<범위>d
이전 동작 취소	u, U	
복사 및 붙이기	yy(#yy), p, P	:#y, :<범위>y, :pu, :#pu
화면 다시 출력	^l	:e!
행 이동	#G, G	:#, :\$
이전 동작 반복	.	
아래 행 이어붙이기	J	
대소문자 바꾸기	~	
문자열 검색		/문자열, ?문자열, n, N
문자열 바꾸기		:s/문자열1/문자열2/g :<범위>s/문자열1/문자열2/g :<범위>s/문자열1/문자열2/gc
다른 파일 읽어오기		:r 파일명
다른 파일 편집으로 가기		:e 파일명, :n
셸 명령 실행		:! 셸명령, :sh

3 버퍼의 사용

vi에서는 버퍼에 복사한 내용을 임시로 저장할 수 있다. 이름이 없는 버퍼를 언네임드 버퍼라고 하고, 이름이 있는 버퍼를 네임드 버퍼라고 한다. 언네임드 버퍼는 한 번에 하나씩만 저장하지만 네임드 버퍼는 버퍼별로 다른 값을 저장할 수 있다. 네임드 버퍼는 “+문자 또는 “+숫자의 형태로 사용한다. 숫자를 사용할 경우 숫자 버퍼라고도 한다.

- 네임드 버퍼 : “a, “b, “c, “d, ..., “z
- 숫자 버퍼 : “1, “2, ..., “9

4 vi 환경설정

명령	기능
:set nu	파일 내용의 각 행에 행 번호 표시(보이기만 할 뿐 저장은 되지 않는다)
:set nonu	행 번호 취소
:set list	눈에 보이지 않는 특수 문자 표시
:set nolist	특수 문자 보기 기능 취소
:set showmode	현재 모드 표시
:set noshowmode	현재 모드 표시 기능 취소
:set	set으로 설정한 모든 vi 변수 출력
:set all	모든 vi 변수와 현재 값 출력



연습문제

- 1 vi에서 이용할 수 있는 세 가지 모드는 무엇인지 간단히 설명하시오.
- 2 마지막 행 모드로 이동하는 키 세 개를 설명하시오.
- 3 행 번호를 표시하는 명령을 기술하시오.
- 4 길이가 5자인 단어를 수정하기 위해 사용할 수 있는 명령키를 나열하시오.
- 5 3행으로 이동할 수 있는 명령은 어떤 것들이 있는지 두 가지 이상 나열하시오.
- 6 파일에 있는 모든 `doday`를 `today`로 고치는 명령키를 기술하시오.
- 7 현재 편집 중인 파일을 `vitest2`라는 이름으로 다시 저장하는 명령을 기술하시오.
- 8 두 행을 한 행으로 만드는 명령을 기술하시오.
- 9 현재 작업 중인 파일에 다른 파일(`vitest3.txt`)을 읽어들이는 명령을 기술하시오.
- 10 파일을 저장하고 vi를 종료하는 명령을 두 개 나열하시오.